NAME: _____ NETID: _____

Exam 2.p.fa.2024

Principles of Computing

11th of December, 2024

UNIVERSITY OF NOTRE DAME

1 *Select the correct response to each of the following questions.*

i. Which of the following would be best described as *"a function that belongs to an object?"*

- ◯ *a lambda*
- ◯ *a class*
- ◯ *a method*
- ◯ *an attribute*

ii. Which of the following methods implements `+` for a class?

- ◯ `__add__`
- ◯ `__init__`
- ◯ `__str__`
- ◯ `__iter__`

iii. Which of the following produces a *syntax error?*

- ◯ `[n for n in range(100) if n >= 50]`
- ◯ `[str(n): n + 1 for n in range(100)]`
- ◯ `[n in range(100)]`
- ◯ `{str(n): n + 1 for n in range(100)}`

iv. Which of the following lists contain exactly 50 elements?

- ◯ `[n for n in range(100) if n >= 50]`
- ◯ `list(range(5)) + list(range(45))`
- ◯ `[n in range(50)]`
- ◯ *all three of the other options*

v. Given an integer list `nice_list`, which of the following returns a new list with the elements squared?

- ◯ `[x**2 for x in nice_list]`
- ◯ `list(map(lambda x: x*x, nice_list))`
- ◯ `list({x: x**2 for x in nice_list}.values())`
- ◯ *all three of the other options*

vi. Which of the following is *not true* about `dict` objects?

- ◯ `dict` *objects have a length*
- ◯ `dict` *objects are mutable*
- ◯ `dict` *objects can be empty*
- ◯ `dict` *objects support iteration*

vii. Which of the following adjectives describes `list` objects but not `tuple` objects?

- ◯ *iterable*
- ◯ *ordered*
- ◯ *mutable*
- ◯ *sortable*

viii. Which of the following keywords is used with *"context managers."*

- ◯ `lambda`
- ◯ `for`
- ◯ `in`
- ◯ `with`

ix. Which of the following types would Santa use for keeping track of which kids are naughty or nice?

- ◯ `dict`
- ◯ `str`
- ◯ `list`
- ◯ `tuple`

x. Which of the following types would Santa use to keep track of the houses he still needs to visit?

- ◯ `dict`
- ◯ `str`
- ◯ `list`
- ◯ `tuple`

NAME: NETID:

2 *Answer the following questions based on the dataset below.*

```
name,valence
Abuela,Nice
Aiko,Naughty
Albert,Naughty
Alberta,Nice
Brody,Nice
Ezra,Nice
Mary-Alice,Nice
Maximus,Nice
Vita,Naughty
Vito,Nice
```

The dataset is in a file whose relative path is given by `data/naughty-or-nice.csv` for each of the following questions.

i. What does the block of code below display in the terminal?

```
with open("data/naughty-or-nice.csv", "r") as file:
  for line in file:
    print(1 if line.strip().split(",")[-1] == "Nice" else 0)
```

.....

ii. What is the *value* of `verdict` after running the code below?

```
judgement = {"Nice": 0, "Naughty": 0}
with open("data/naughty-or-nice.csv", "r") as file:
  next(file)
  for line in file:
    judgement[line.strip().split(",")[1]] += 1
verdict = sum(judgement.values())//len(judgement)
```

.....

iii. What is the *value* of `present` after running the code below?

```
info = {}
with open("data/naughty-or-nice.csv", "r") as file:
  for line in list(file)[1:]:
    elf = line.split(",")[0][0]
    info[elf] = 1 if elf not in info else info[elf] + 1
present = max(info.keys(), key=lambda x: info[x])
```

.....

iv. What is the *value* of `fireplace` after running the code below?

```
with open("data/naughty-or-nice.csv", "r") as file:
  next(file)
  log = [line.strip().split(",") for line in file]
fire = list(map(lambda x: len(x[0]) + len(x[1]), log))
fireplace = max(enumerate(fire), key=lambda x: x[1])[0]
```

.....

NAME: NETID:

3 *Answer the following questions based on the class below.*

```
class Point:
  def __init__(self, x, y):
    self.x = x
    self.y = y
  def __eq__(self, point):
    if isinstance(point, Point):
      return (self.x == point.x) and (self.y == point.y)
    if isinstance(point, tuple) or isinstance(point, list):
      return self.x == point[0] and self.y == point[1]
  def __add__(self, point):
    return Point(self.x + point.x, self.y + point.y)
  def __sub__(self, point):
    return Point(self.x - point.x, self.y - point.y)
  def __mult__(self, other):
    if isinstance(other, int) or isinstance(other, float):
      return Point(scalar*self.x, scalar*self.y)
    elif isinstance(other, Point):
      return self.x*other.x + self.y*other.y
  def __str__(self):
    return f"({self.x}, {self.y})"
```

i. What does the following block of code display in the terminal?

```
snowball = Point(0, 0)
trajectory = Point(1, 1)
for i in range(5):
  snowball = snowball + trajectory
  trajectory = trajectory - (0, 0.1)
print(snowball)
```

.....

ii. What does the following block of code display in the terminal?

```
rudolf = Point(0, 0)
rudolf.x, rudolf.y = (5, 2)
print(rudolf*rudolf)
```

.....

iii. The code below throws an error; on *what line* does it occur?

```
north_pole = (0, 0)
santa = Point(64, 32)
cookies =
milk =
while not santa == north_pole:
  cookies = santa.x + santa.y
  milk = (santa*2).y - santa.x
  santa = santa // 2
```

.....

NAME: NETID:

4 *Program a solution to the following question in Python.*

You are given a dataset `map.csv` formatted according to the rules below.

1. The first line is a string consisting of the characters `"L"` and `"R"`.
2. The second line is a *token* representing a *starting location*.[1]
3. Every line afterwards consists contains three *tokens:*
   (a) The first token represents a *start* location.
   (b) The second and third tokens represent *left* and *right* destinations.
   (c) The first token is separated from the other tokens by `" = "`.
   (d) The second and third tokens are surrounded by parentheses.
   (e) The second and third tokens are separated by `", "`.
   (f) First tokens will never be repeated.

These lines represent transitions An example dataset is given below.

```
LRRL
AAA
AAA = (BBB, CCC)
BBB = (DDD, EEE)
CCC = (ZZZ, GGG)
DDD = (DDD, EEE)
EEE = (BBB, ZZZ)
GGG = (GGG, GGG)
ZZZ = (ZZZ, ZZZ)
```

Your task is traverse the map beginning at the starting location[2] by following the `"L"` and `"R"` directions[3] and seeing where your traversal ends. In the directions, an `"L"` signifies that you should go to the *second* token,[4] and an `"R"` signifies that you should go to the *third* token.[5] In the example dataset above, our final destination would be `ZZZ`.

`AAA` $\xrightarrow{\text{L}}$ `BBB` $\xrightarrow{\text{R}}$ `EEE` $\xrightarrow{\text{R}}$ `ZZZ` $\xrightarrow{\text{L}}$ `ZZZ`

Write a block of code that reads this file and prints the final destination of reached by traversing the map according to the directions.

RESTRICTIONS:
- No use of `import` statements.
- No use of functions, methods, nor types that we have not covered in class nor problem sets.

[1] A *token* is a string of three consecutive upper-case letters.

[2] given by the token on the second line

[3] given by the string on the first line

[4] the token on the *left* of the ordered pair

[5] the token on the *right* of the ordered pair

NOTE: you may name your function and your variables whatever you'd like.